TF-IDF algorithm based on term count distribution information and category distribution information.

Problem Statement:

TF-IDF feature weight calculation algorithm requires term count as a measure of the criticality of features. Text categorization includes text pre-processing, feature extraction, classifier selection and classifier evaluation. Text term classification transforms text information into feature vectors, any term in the training data might be a feature term, that makes the featured-vector dimensionality very high and affects the classification results.

Most used feature space dimensionality reduction method feature extraction is based on calculating the weight of features and removing terms with lower weights to reduce dimension. Feature extraction method can be divided into the following categories according to the difference of feature evaluation functions:

(1) Document Count (DF): The count of the document containing the feature term in the text data set is used as the measure for evaluation.
When the feature term primarily exists in a certain type of document, it should be possible to reflect in the text feature through the feature term but is eliminated due to the document count is less than a given threshold, resulting in poor classification effect.

(2) Information Gain (IF): This feature measure of evaluation refers to the difference between the information entropy of the feature term and the information entropy of the feature term that is not included in the text.
The information gain method affects the text classification decision because of considering the case that the feature terms are not present.

(3) Chi-square Statistic (CHI): Measures weight of relationship between the feature term and the category as a measure of evaluation to measure the criticality of the feature.
Chi-square statistics is relatively stable, but the feature terms with low document counts gets selected minimizing the classification accuracy when feature terms and relative categories do not meet the chi-square distribution criteria.

(4) Term Count-Inverse Document Count (TF-IDF): The feature weight depends on the degree of the term's contribution to the document containing the term.
TF-IDF algorithm does not consider the difference between the text categories of the data set and the distribution of text vectors within the categories, hence classification impact on the imbalanced dataset is not optimized.

TF-IDF algorithm is commonly used because of simplicity but it has limitations as below.
(1) Imbalance in the distribution of dataset categories is not considered which is mostly uneven, and often has certain differences. When the classification of data sets is different, the calculated IDF is smaller ultimately affecting the classification accuracy.

(2) There is no correct response to the distribution of text vectors between classes. The inter class distribution of text vectors needs to be considered when the feature term has a high count in the class and a low count in other classes.

Solution Approach:

Paper suggests feature weight calculation algorithm FDCD-TF-IDF based on term count distribution information and category distribution information. This algorithm introduces the idea of term count distribution and class distribution to describe the weight of the feature term more accurately. The term count distribution is mainly aimed at the correlation between feature terms and categories, and the category distribution can better reflect category information of feature terms.

TF-IDF is a feature weighting algorithm that merges term count weight and inverse document count to compute the weight of feature terms in text.

TF: refers to the count of the feature terms appearing in a text, and the feature terms with higher count are given larger weights, and the features with less count are given lower weights.

$$tf_{i,j}=n_{i,j} / \sum_k n_{k,j}$$

$n_{i,j}$ is the number of times in the feature term ti in document dj,
$\sum_k n_{k,j}$ is the sum of the appearance times of all the terms in the document d_j.

IDF: refers to the measure of the degree of separation by a category of a term. For the term t_i., its IDF can be divided by the number of documents containing t_i by the number of documents N of the entire data set, and then the logarithm of its quotient is obtained. The calculation formula is:

$$idf_{i,=} \log(N/n)$$

the denominator is usually added to 1.

TF-IDF: the feature extraction algorithm merges the term count TF and the inverse document count IDF, and its feature extraction function is:

$$f(t_{i)=} tf_{i,j} * idf_i = tf_{i,j} * \log(N/(n+1))$$

FDCD-TF-IDF algorithm.
In imbalanced datasets with category distribution, standard feature selection algorithms generally prefer to select feature terms in large categories with the difference of term count distribution between inter-class and intra-class will result in weights of final features.
Standard TF-IDF algorithm doesn't deal with the imbalanced datasets well and considers the term count distribution information between inter-class and intra-class at the same time. This paper suggests algorithm FDCD-TF-IDF based on term count distribution information and category distribution information of TF-IDF.

σ : Inter-class constant, reflects the inter-class distribution information of feature terms obtained by calculating the logarithm of the quotient between the number of documents a_i containing the feature term t_i in the C_j class and the number of documents c_i of the non-C_j class containing the feature term t_i.

$$\sigma = \log_2(2 + (a_i / (C_i+1)))$$

ψ**:** Intra-class distribution factor, that reflects the intra-class distribution information of feature terms. It can be obtained by calculating the logarithm of the quotient between the number of documents a_i in the C_j class that contains the feature term t_i and the number of documents b_i without the feature term t_i in the C_j class. The specific formula is:

$$\psi= \log_2(2 + (a_i / (B_i+1)))$$

ω **:** Category distribution factor, that reflects the distribution information of document categories. It can be obtained by calculating the logarithm of the quotient between the total number of document N in the dataset and the number of all documents n_j in the C_j class. Its calculation formula is:

$$\omega = \log_2(n_{i+}Ni)$$

Therefore, formula for calculating the weight of FDCD-TF-IDF is:

$$FDCD - TF - IDF = tf_{i,j} * idf_i * \sigma * \psi * \omega$$

Ratio of high inter-class distribution σ indicates that the distribution of feature term is higher in the C_j class, and is less in the non- C_j class. Correlation degree between the feature term t and the C_j class is larger, and the weight value should be higher.

When the ratio of the intra-class distribution factor ψ is larger, it means the contribution of the category is greater.

Class distribution factor ω is mainly used to adjust the categorical factors to a certain extent so that they are treated with the same attention as the small categories in the feature selection process.

Results:

The text corpus of Sogou laboratory, and the classifier is linear SVM. The classification result of FDCD-TF-IDF algorithm is compared with standard TF-IDF algorithm, and f1-score is used as the evaluation indicator of classification performance.

class-balanced Dataset.

Document category	TF-IDF+ SVM_linear	FDCD-TF-IDF + SVM_linear
	F1 Score	**F1 Score**
Finance	88.500%	91.940%
IT	86.961%	90.066%
health	91.975%	94.833%
sports	95.413%	98.795%
tourism	91.522%	96.250%
education	89.857%	92.000%
recruitment	86.073%	87.948%
culture	82.096%	87.948%
military	92.161%	97.403%
all	89.395%	93.020%

class-unbalanced Dataset.

Document category	TF-IDF+ SVM_linear	FDCD-TF-IDF + SVM_linear
	F1 Score	**F1 Score**
Finance	70.588%	83.846%
IT	67.588%	84.354%
health	75.788%	89.189%
sports	71.000%	91.683%
tourism	78.873%	89.333%
education	83.047%	86.769%
recruitment	63.636%	84.286%
culture	68.657%	81.366%
military	92.161%	96.447%
all	74.593%	87.475%

For the class-balanced datasets it is seen the classification results are increased by 3.625% showing algorithm FDCD-TF-IDF can accurately calculate the feature term weighting on the class-balanced datasets.

For the class-unbalanced dataset it is seen from the table that the classification result of algorithm FDCD-TF-IDF has accuracy 12.882% higher than the original algorithm.

The experimental results show that the term count distribution information and category distribution information introduced by the superior algorithm are reasonable, and the validity of the superior algorithm is verified.

Conclusion:

In this paper superior algorithm FDCD-TF-IDF based on TF-IDF is proposed by introducing the idea of term count distribution and category distribution. The experimental results show that the algorithm can achieve better classification result in the class-balanced and class-unbalanced datasets, showing that proposed algorithm is reasonable and effective.

Identifying the limitations of the current paper and suggesting the Improvements:

limitations of the current paper:

Paper discusses in great details of how the method is effective for class imbalanced data and also shows with results that suggested method FDCD-TF-IDF is truly ideal for class unbalanced data where F1 scores are boosted by 12.882% as compared to normal TF-IDF data.
Paper tries to justify the use of Method FDCD-TF-IDF in class balanced data with increase in 3.625% as compared to normal TF-IDF data.

However, paper does have below issues.

- Paper does not discuss additional computing cost that will be required to compute FDCD-TF-IDF as applicable to real life scenarios.
- Results achieved with standard TF-IDF method for class balanced data are comparable with FDCD-TF-IDF and when compared with computing cost gains are limited and method will be suitable only in cases where the cost of computing is secondary to achieving higher results when already achieved accuracy of TF-IDF class is higher.
  Which in case of given exercise of classification of document is unlikely scenario unless documents are of scientific grades as such efforts are not applicable for business class scenarios where cost incurred is higher compared to returns.
- FDCD-TF-IDF has shown significant improvement are the classes where already achieved F1 score is below certain threshold. Specifically, document having F1 score above 80% with TF-IDF does not show a great improvement.
  This is because of fact that terms used in such classes are already utilised correctly to achieve the given accuracy irrespective of whether the data is balanced or imbalanced.

Suggesting the Improvements:
Proposed Methods:

- Candidate documents are needed to be identified beforehand of proceeding for either TF-IDF and FDCD-TF-IDF scenarios.
- To come up with technique which gives better overall results compared to standard TF-IDF but is low cost compared to FDCD-TF-IDF

Method 1:
Candidate documents are needed to be identified beforehand of proceeding for either TF-IDF and FDCD-TF-IDF scenarios:
Suggested is to carry out chi squared tests beforehand finding candidate CORPUS for TF-IDF or FDCD-TF-IDF documents with selected terms or features aligned with TF-IDF document. This will give us impact of features over a class.

$$\lambda^2 = \sum (O - E)^2 / E$$

where O is observed value and E is estimated value.

Significant Features.

Feature	Class1	Class2
F1	Observed, expected	Observed, expected
F2	Observed, expected	Observed, expected
F3	Observed, expected	Observed, expected
F4	Observed, expected	Observed, expected
F5	Observed, expected	Observed, expected
F6	Observed, expected	Observed, expected
F7	Observed, expected	Observed, expected

We could determine the threshold for numbers of features that could be selected for creating vector for doing TF-IDF analysis.
Now for all such selected feature terms if we are able to determine threshold in terms of overall impact on a class these selected features are making, then from this threshold we could approximate TF-IDF score will be below a certain threshold or not. In case if it is below that threshold, we could say that we will use TF-IDF otherwise we will use FDCD-TF-IDF analysis.

TF-IDF candidate Features.

Feature	Class1	Class2
F1	Observed, expected	Observed, expected
F2	Observed, expected	Observed, expected
F3	Observed, expected	Observed, expected
F4	Observed, expected	Observed, expected

Comparing λ^2 Values for features and TF-IDF candidate features we could determine if there is significant improvement beyond certain threshold, we could say we will use TF-IDF to determine or else FDCD-TF-IDF to predict document class.

Given a low cost of chi square test if we select significant features vs TF-IDF candidate features chi square test at low cost will be able to determine if we should further opt for TF-IDF or FDCD-TF-IDF.

Method 2:
To come up with technique which gives better overall results compared to standard TF-IDF but is low cost compared to FDCD-TF-IDF

One such method can be directly said as entropy boosting
We first try to identify ways of measuring the term's relationship with different classes. Then we see how this information can be leveraged to assign new data to appropriate classes. For terms that are inherently specific to certain classes, we expect their distributions to be concentrated in those classes. On the other hand, terms that are generic gets roughly uniformly distributed over all the classes. One common way to identify presence or absence of such concentrations is through Entropy. We could compute the entropy.
.

Normalization of Entropy Boosting

Once we calculated the entropy of each term in the corpus, we want to get an estimate of how informative a term is for each class. We proposed an entropy-based approach called Normalized Entropy Boosting. We may calculate Normalized Entropy as

$$NE = (\text{Entropy}_{max} - \text{Entropy calculated}) / \text{Entropy}_{max}$$

Terms that are concentrated in few classes should have higher NE whereas terms that are almost uniformly distributed among classes should have lower NE. Although TF. IDF_{NE} gives better accuracy than traditional TF-IDF and will be less costly than proposed FDCD-TF-IDF.